

Παράλληλος Προγραμματισμός 2019

Προγραμματιστική Εργασία #1

Ονοματεπώνυμο: Ζαχαριάδης Αλέξανδρος

A.M.: Π2015082

A. Περιγραφή του Κώδικα

Στο πρόγραμμα **matmul-normal.c** γίνεται ο πολλαπλασιασμός των πινάκων χωρίς εντολές **SSE2**. Στην αρχή υπάρχει η συνάρτηση **get_walltime** που χρησιμοποιείται για να πάρουμε το χρόνο στο τέλος. Στη συνέχεια, γίνεται η δήλωση των μεταβλητών που χρειάζονται και η δέσμευση μνήμης για τους πίνακες με τη χρήση της συνάρτησης **malloc**. Σε περίπτωση που μία δέσμευση αποτύχει, τυπώνεται ένα μήνυμα, ελευθερώνονται οι θέσεις μνήμης που έχουν ήδη δεσμευτεί και τερματίζει το πρόγραμμα. Ύστερα, γίνεται η αρχικοποίηση των πινάκων και ξεκινάει η μέτρηση του χρόνου με την εντολή **get_walltime(&ts)**. Μέσα στο φορτίο, κάνουμε τον pointer **pc** να δείχνει στον πίνακα **c** και φτιάχνουμε μία **for** που θα διαπερνάει τις στήλες (ή γραμμές αφού θεωρούμαι ότι έχει γίνει αντιμετάθεση γραμμών-στηλών) του πίνακα **b**. Μετά κάνουμε τον **pb** να δείχνει στον πίνακα **b** και βάζουμε άλλη μία **for** για τις γραμμές του **a**. Γράφοντας **pa = a + k*N** κάνουμε τον **pa** να δείχνει στη σωστή γραμμή κάθε φορά. Στη συνέχεια μηδενίζουμε το **sum** και με μία ακόμα **for** πολλαπλασιάζουμε τη στήλη του ενός πίνακα με τη γραμμή του άλλου. Όταν τελειώσει ο πολλαπλασιασμός, αναθέτουμε την τιμή του **sum** στο σημείο που δείχνει ο pointer **pc** και τον αυξάνουμε για να δείχνει στην επόμενη θέση μνήμης. Στο τέλος του φορτίου μετράμε το τέλος του χρόνου με την εντολή **get_walltime(&te)**. Μετά ελέγχουμε αν έγινε σωστά ο πολλαπλασιασμός και στην περίπτωση που βρεθεί κάποιο λάθος, τυπώνεται ένα μήνυμα και τερματίζει το πρόγραμμα. Αν είναι όλα σωστά, τότε τυπώνεται ο χρόνος του φορτίου, αποδεσμεύονται οι πίνακες και το πρόγραμμα τερματίζει ομαλά.

Στο πρόγραμμα **matmul-sse.c** γίνεται ο πολλαπλασιασμός των πινάκων με τη χρήση εντολών **SSE2**. Όλο το πρόγραμμα εκτός από το φορτίο, είναι ίδιο αλλά με τη διαφορά ότι η δέσμευση των πινάκων γίνεται με την **posix_memalign** για να γίνει η δέσμευση σε συνεχόμενες θέσεις μνήμης. Στο φορτίο, δίνουμε στη μεταβλητή **vproduct** (τύπου **__m128**) τη διεύθυνση μνήμης του πίνακα **product**, ο οποίος έχει μέγεθος 4 θέσεων για float επειδή εκεί θα αποθηκεύονται τα αποτελέσματα των **intrinsics** της μορφής **_mm_????_ps**. Επίσης, κάνουμε το δείκτη **pc** να δείχνει στον πίνακα **c**. Ύστερα, παρόμοια με το άλλο πρόγραμμα, κάνουμε μία **for** για τις στήλες του **b**, κάνουμε το **vb** να δείχνει σε αυτόν και βάζουμε άλλη μία

for για τις γραμμές του **a**. Μετά κάνουμε το **vb** να δείχνει στη σωστή γραμμή του **a**, μηδενίζουμε το **sum** και βάζουμε άλλη μία **for**. Επειδή με τη χρήση των **intrinsics** οι πράξεις γίνονται ανά τετράδες, η **for** προχωράει ανά 4 στοιχεία. Στο **vproduct** αποθηκεύονται οι τετράδες του πολλαπλασιασμού με την εντολή ***vproduct = _mm_mul_ps(*va, *vb)** και στη συνέχεια αυτές οι τετράδες προστίθενται στο **sum** για να υλοποιηθεί σωστά ο πολλαπλασιασμός των πινάκων. Τέλος, μετά την εσωτερική **for**, αναθέτουμε την τιμή του **sum** στο σημείο που δείχνει ο pointer **pc** και τον αυξάνουμε για να δείχνει στην επόμενη θέση μνήμης.

Για να γίνει σίγουρο ότι ο μεταγλωττιστής δεν απαλείφει τα loop χρησιμοποιούνται αντίστοιχα στο command line οι εντολές:

- **gcc -O2 matmul-normal.c -S -DN=4**
- **gcc -O2 -msse2 matmul-sse.c -S -DN=4**

Για να τρέξουμε τα προγράμματα γράφουμε αντίστοιχα στο command line:

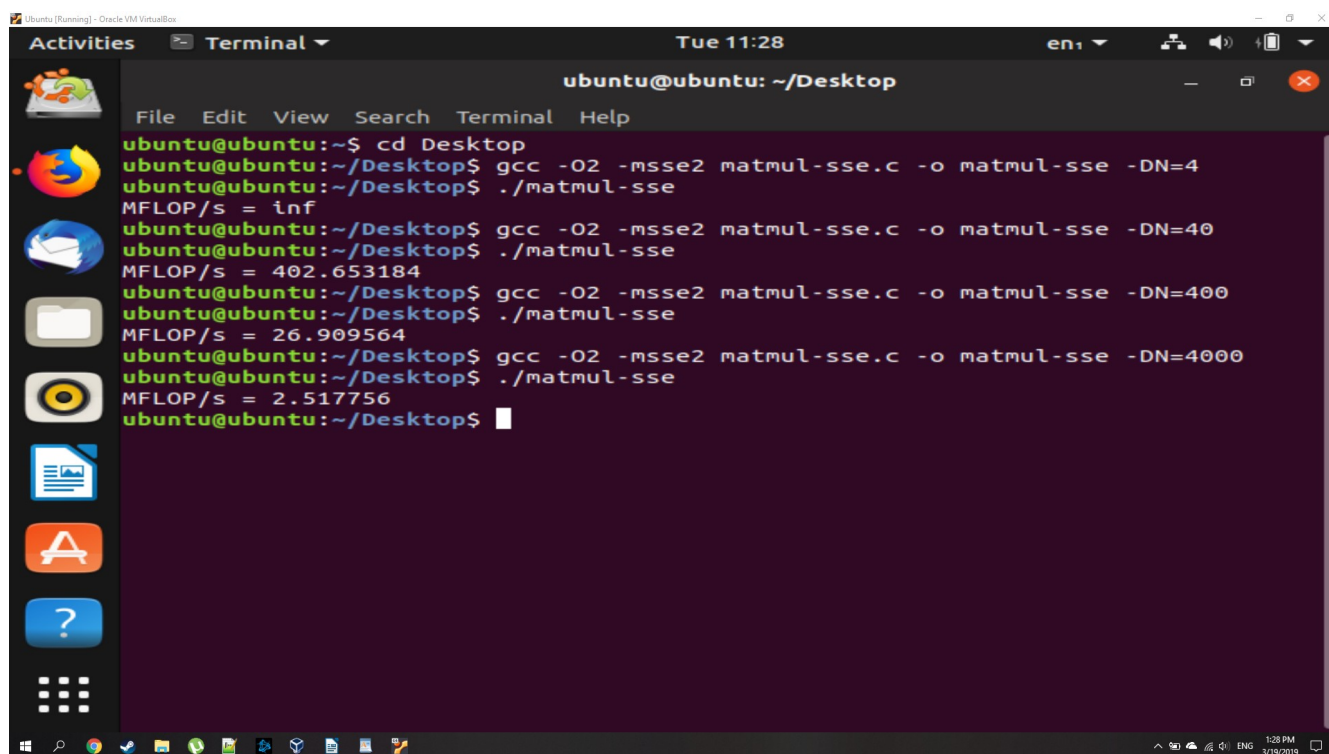
- **gcc -O2 matmul-normal.c -o matmul-normal -DN=4** και μετά **./matmul-normal**
- **gcc -O2 -msse2 matmul-sse.c -o matmul-sse -DN=4** και μετά **./matmul-sse**

B. Πίνακας Αποτελεσμάτων

	4	40	400	4000
matmul-normal.c	inf	204.392479	11.622394	1.153917
matmul-sse.c	inf	402.653184	26.909564	2.517756

```

ubuntu@ubuntu: ~/Desktop
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ cd Desktop
ubuntu@ubuntu:~/Desktop$ gcc -O2 matmul-normal.c -o matmul-normal -DN=4
ubuntu@ubuntu:~/Desktop$ ./matmul-normal
MFLOP/s = inf
ubuntu@ubuntu:~/Desktop$ gcc -O2 matmul-normal.c -o matmul-normal -DN=40
ubuntu@ubuntu:~/Desktop$ ./matmul-normal
MFLOP/s = 204.392479
ubuntu@ubuntu:~/Desktop$ gcc -O2 matmul-normal.c -o matmul-normal -DN=400
ubuntu@ubuntu:~/Desktop$ ./matmul-normal
MFLOP/s = 11.622394
ubuntu@ubuntu:~/Desktop$ gcc -O2 matmul-normal.c -o matmul-normal -DN=4000
ubuntu@ubuntu:~/Desktop$ ./matmul-normal
MFLOP/s = 1.153917
ubuntu@ubuntu:~/Desktop$
  
```



```
ubuntu@ubuntu: ~/Desktop
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ cd Desktop
ubuntu@ubuntu:~/Desktop$ gcc -O2 -msse2 matmul-sse.c -o matmul-sse -DN=4
ubuntu@ubuntu:~/Desktop$ ./matmul-sse
MFLOP/s = inf
ubuntu@ubuntu:~/Desktop$ gcc -O2 -msse2 matmul-sse.c -o matmul-sse -DN=40
ubuntu@ubuntu:~/Desktop$ ./matmul-sse
MFLOP/s = 402.653184
ubuntu@ubuntu:~/Desktop$ gcc -O2 -msse2 matmul-sse.c -o matmul-sse -DN=400
ubuntu@ubuntu:~/Desktop$ ./matmul-sse
MFLOP/s = 26.909564
ubuntu@ubuntu:~/Desktop$ gcc -O2 -msse2 matmul-sse.c -o matmul-sse -DN=4000
ubuntu@ubuntu:~/Desktop$ ./matmul-sse
MFLOP/s = 2.517756
ubuntu@ubuntu:~/Desktop$
```

Γ. Εξήγηση Αποτελεσμάτων

Στο *matmul-normal.c* και στο *matmul-sse.c* για $N=4$ το αποτέλεσμα βγαίνει άπειρο επειδή οι πράξεις γίνονται πολύ γρήγορα σε αυτόν τον υπολογιστή. Για $N=40$ το πρόγραμμα χωρίς τις εντολές **SSE2** βγάζει σαν αποτέλεσμα γύρω στα 200 MFLOP/s σε αντίθεση με αυτό που χρησιμοποιεί **SSE2** που βγάζει 400 MFLOP/s. Επίσης, για $N=400$ και $N=4000$ χωρίς **SSE2** έχω 11.6 και 1.5 MFLOP/s, ενώ με **SSE2** έχω 26.9 και 2.5 MFLOP/s αντίστοιχα. Φαίνεται λοιπόν ότι με τη χρήση των εντολών **SSE2** έχουμε περίπου δύο φορές καλύτερα αποτελέσματα. Αυτό γίνεται επειδή, στο δεύτερο πρόγραμμα οι πράξεις γίνονται ανά τετράδες με τη χρήση των intrinsics τα οποία είναι ειδικά διαμορφωμένα για χρήση από compilers. Επίσης, το λειτουργικό σύστημα διαθέτει ειδικούς καταχωρητές για **SSE** και έτσι οι μεταβλητές `__m128` αποθηκεύονται εκεί πέρα αυξάνοντας ακόμα περισσότερο την απόδοση.